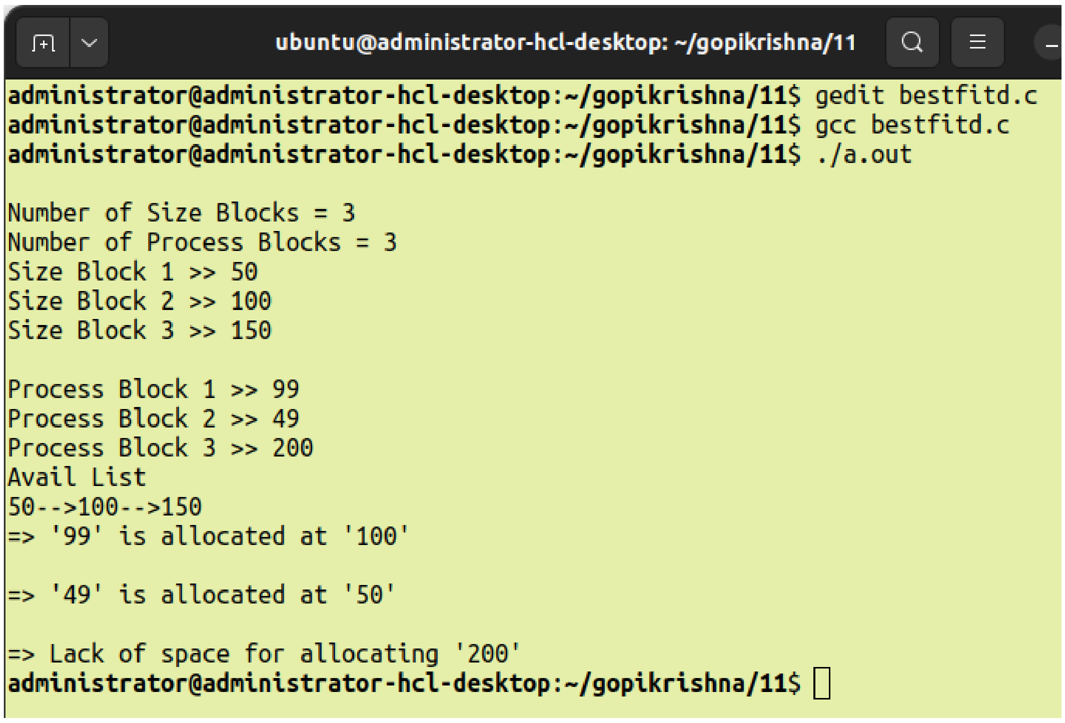
```
/*
            GOPIKRISHNA V
            S3 CSE A
            52
*/
 #include<stdio.h>
 #include<stdlib.h>

 struct node
 {
    int data;
    struct node *n_link;
 };

 struct node *avail = NULL,*current_node, *new_node, *prev, *temp, *smallest_add;
 int bal = 0, smallest = 0, best, count = 0;
 int n_blocks, n_process, block, process;

 struct node *get_node(int ele)
 {
    temp = (struct node *)malloc(sizeof(struct node));
    if(temp == NULL)
       return NULL;
    else
    {
       temp->data = ele;
       temp->n_link = NULL;
    }
    return temp;
 }

 void insert(int ele)
 {
    new_node = get_node(ele);
    if(new_node != NULL)
    {
       if(avail == NULL)
          avail = new_node;
       else
       {
          current_node = avail;
          while(current_node->n_link != NULL)
          {
             current_node = current_node->n_link;
          }
          current_node->n_link = new_node;
       }
    }
    else
    {
    printf("No Node Created");
    }
```

```
}

void display()
{
    printf("Avail List\n");
    current_node = avail;

    while(current_node != NULL)
    {
        printf("%d", current_node->data);
        current_node = current_node->n_link;
        if(current_node != NULL)
        {
            printf("-->");
        }
    }
}

void delete(struct node *address)
{
    prev = avail;
    current_node = prev->n_link;

        while(current_node != NULL && current_node != address)
        {
            count++;
            current_node = current_node->n_link;
            prev = prev->n_link;
        }
        if(current_node != NULL)
        {
            prev->n_link = current_node->n_link;
            free(current_node);
        }
        else if(current_node == NULL && count == 0)
        {
            avail = NULL;
            free(current_node);
        }
        else if(current_node == NULL)
        {
            prev->n_link = NULL;
            free(current_node);
        }
        count=0;
}

int allocate(int process)
{
 current_node = avail;
 smallest = 10000;
 best = 0;
 while(current_node != NULL)
 {
    bal = current_node->data-process;
    if(smallest > bal && bal >= 0)
    {
        smallest = bal;
        smallest_add = current_node;
        best = current_node->data;
    }
    current_node = current_node->n_link;
 }
 if(smallest_add == avail)
 {
     avail = smallest_add->n_link;
     free(smallest_add);
 }
 else
   delete(smallest_add);
 return best;
}

void main()
{
    printf("\nNumber of Size Blocks = ");
    scanf("%d", &n_blocks);
    LABEL:
        printf("Number of Process Blocks = ");
        scanf("%d", &n_process);
    int a[n_process];
    if(n_process>n_blocks)
    {
        printf("Only %d Size Blocks Available\n",n_blocks);
        goto LABEL;
    }
    else
    {
        for(int i=1;i<=n_blocks;i++)
        {
            printf("Size Block %d >> ",i);
            scanf("%d",&block);
            insert(block);
        }
    }
    printf("\n");
    for(int i=1;i<=n_process;i++)
    {
        printf("Process Block %d >> ",i);
        scanf("%d",&process);
        a[i]=process;
    }
    display();
    for(int i=1;i<=n_process;i++)
    {
        int best_space=allocate(a[i]);
        if(best_space==0)
            printf("\n=> Lack of space for allocating '%d'\n",a[i]);
        else
            printf("\n=> '%d' is allocated at '%d'\n",a[i],best_space);
    }
}
```

OUTPUT



```
administrator@administrator-hcl-desktop:~/gopikrishna/11$ gedit bestfitd.c
administrator@administrator-hcl-desktop:~/gopikrishna/11$ gcc bestfitd.c
administrator@administrator-hcl-desktop:~/gopikrishna/11$ ./a.out

Number of Size Blocks = 3
Number of Process Blocks = 3
Size Block 1 >> 50
Size Block 2 >> 100
Size Block 3 >> 150

Process Block 1 >> 99
Process Block 2 >> 49
Process Block 3 >> 200
Avail List
50-->100-->150
=> '99' is allocated at '100'

=> '49' is allocated at '50'

=> Lack of space for allocating '200'
administrator@administrator-hcl-desktop:~/gopikrishna/11$ 
```